REMARKS

It is believed that the above amendments and following remarks attend to each and every rejection and objection presented in the July 26, 2005 Office Action. Claims 1-19 remain pending, with claims 1 and 14 being independent.

Drawing Objections

The drawings stand objected to under 37 CFR § 1.83(a) for failing to include urgency indicators. A replacement sheet containing FIG. 1 and FIG. 2 is attached. FIG. 1 is amended to show Urgency Indicators 32. Support for this amendment is at least provided by paragraphs [0006], [0007] and [0020]. For example, paragraph [0006] teaches, "each thread may include an urgency identifier." Paragraph [007] teaches that urgency is an abstraction of how well a thread is progressing (or will progress) within a pipeline. Paragraph [0020] teaches, "controller 30 may also modify the urgency of a thread" Accordingly, Urgency indicators are illustratively shown within thread controller 30. FIG. 1 is further amended to correct a typographical error. Register file 16(N) is corrected to read register file 16(M), since the register files map directly to threads 15 and not pipelines.

Specification Amendment

In accord with the drawing amendment, paragraph [0016] is amended to describe the urgency indicators now illustratively shown within FIG. 1. Paragraph [0016] is further amended to correct typographical error; the total number of register files is 'M' and not 'N', since the register files map directly to threads 15. Paragraph [0016] is again amended to remove an extraneous parenthesis near the introduction of thread 15(M). No new matter is added.

Claim Rejections – 35 U.S.C. §102

Claims 1-19 stand rejected under 35 U.S.C. §102(b) as being anticipated by Concurrent Event Handling through Multithreading, 1999, IEEE Transactions on Computers, volume 48, NO. 9, pages 903-916 (hereinafter "Keckler"). Respectfully, we disagree. To anticipate a claim, Keckler must teach every element of the claim and "the identical invention must be shown in as complete detail as contained in the ... claim." MPEP 2131 citing Verdegaal Bros. V. Union Oil Co. of California, 814 F.2d

628, 2 USPQ2d 1051 (Fed. Cir. 1987) and Richardson v. Suzuki Motor Co., 868 F.2d 1226, 9 USPQ2d 1913 (Fed. Cir. 1989). Keckler does not teach every element of claims 1-19.

Claim 1 recites a method for determining thread switch points within pipeline execution units of a processor, including:

a) monitoring instruction processing of a first thread within the pipeline execution units;

b) in the event of a possible switch point within the pipeline execution units, deactivating the first thread, or not, based upon a first urgency indicator for the first thread, the first urgency indicator being based upon progress of the first thread within the pipeline execution units.

An 'urgency' for the thread is for example based upon "how well a thread is progressing (or will be progressing) within a pipeline" and "how urgent the program or processor logic believes the thread should be". See Applicant's specification, paragraph [0007]. "An assessment is also made of a thread's execution and relative to time-slice expiration." See Applicant's specification, paragraph [0009].

Keckler, on the other hand, discloses a processing chip that "makes its thread selections based upon the availability of an instruction's operands." See Keckler page 908, section 3.2. Keckler also does not disclose or suggest an urgency indicator upon which deactivation of a thread is based. In paragraph 5 of the pending office action, the Examiner asserts that indication of all instruction data arriving is based upon the progress of the thread within the pipeline execution units. Respectfully we disagree. Keckler shows no relationship between an instruction's operand availability and progress of the thread through a pipeline. Rather, Keckler discloses that "if an instruction's input data is not available, the instruction will not execute. Thus the unavailability of an instructions operands **causes** the pipeline to stall, and is clearly not an indicator of a thread's progress through the pipeline. As known in the art, operand availability is based upon latency of register and memory access and is not based upon a thread's progress through a pipeline. Teaching away from claim 1, Keckler instead discloses a "simple three state priority scheme" that "enables critical threads ... to use a higher fraction of the execution resources..." See Keckler, page 908, section 3.2.1. The priority system of Keckler is static and depends upon the

application of the thread (i.e., a user thread, a system thread, etc). It is not based upon progress of the thread within the pipeline execution units as required by claim 1.

Keckler does not disclose an urgency indicator and therefore cannot disclose or suggest 'deactivating the first thread, or not, based upon a first urgency indicator for the first thread' as required by claim 1. Reconsideration of claim 1 is requested.

Claims 2-13 depend from claim 1 and benefit from like arguments; but in addition these claims have other features that patentably distinguish over Keckler. For example, claim 2 recites deactivating the first thread and activating a second thread based upon a second urgency indicator for the second thread, the second urgency indicator being based upon expected progress of the second thread within the pipeline execution units. Keckler does not disclose or suggest using a second urgency indicator of a second thread to decide upon deactivating the first thread and activating the second thread, as required by claim 2.

Claim 3 recites deactivating the second thread, or not, based upon the second urgency indicator for the second thread and in the event of a possible switch point event of the second thread. Keckler does not disclose deactivating the second thread, or not, based upon the second urgency indicator for the second thread and in the event of a possible switch point event of the second thread, as required by claim 3.

Claim 4 recites activating another thread within the pipeline if the second thread is switched out. Keckler does not disclose or suggest activating another thread within the pipeline if the second thread is switched out, as in claim 4.

Claim 5 recites deactivating the first thread, or not, based upon the first urgency indicator and upon a second urgency indicator of a second thread, the second urgency indicator being based upon expected progress of the second thread within the pipeline execution units. Keckler does not disclose an urgency indicator being based upon expected progress of a thread.

Claim 6 recites utilizing a thread controller coupled with the execution units. The Examiner asserts that the Synchronization stage of Keckler represents the thread controller coupled with the execution units. Respectfully we disagree. Specifically, claim 6 requires that the thread controller monitors instruction processing of a first thread within the pipeline execution units. Keckler's synchronization (SZ) stage, on the other hand, is a dedicated pipeline stage. Keckler discloses "one 3-wide

instruction waits in the SZ stage's reservation station until the instruction is ready to issue." See Keckler section 3.2.1, page 908. Clearly, Keckler's synchronization stage operates upon instructions, and does not monitor instruction processing of a first thread.

Claim 7 recites modifying the first urgency indicator to increase or alternatively decrease urgency of the first thread based upon characteristics associated with the possible switch point. Keckler does not teach or suggest modifying an urgency indicator based upon associated possible switch point.

Claim 8 recites determining whether a time slice expiration occurred. Keckler does not disclose determining whether a time slice expiration occurred. In fact, teaching away from claim 8, Keckler states "the MAP chip makes its thread selections based upon the availability of an instructions register operands." See Keckler page 908, section 3.2. The Examiner asserts that the preempt state of Keckler is equivalent to time slicing. Respectfully, we disagree. The preempt state of Keckler guarantees "that a ready instruction can only be stalled for up to 255 cycles." See Keckler page 908, section 3.2.1. As known in the art, time slicing provides a method of ensuring equal periods of processor utilization between threads. We contend that the preempt state of Keckler is not equivalent, at lease because this preempt state does not divide processor utilization equally between threads, and is only used when "one thread is creating a tremendous number of events," such that "other threads will be able to continue making forward progress." See Keckler pages 908-909, end of section 3.2.1.

Claim 9 recites utilizing a time slice expiration unit. As argued above, the preempt state of Keckler is not equivalent to time slicing, and therefore Keckler's preempt state cannot represent a time slice expiration unit.

Claim 10 recites determining whether a cache miss occurred. Again, teaching away from claim 10, Keckler states "Unlike block multithreading ... which switches threads on long latency operations, such as cache misses ... the MAP chip makes its thread selections based upon the availability of an instructions register operands." See Keckler page 908, section 3.2.Thus, Keckler has no need to detect cache misses, since operation of the MAP chip does not depend upon them.

Claim 11 recites inserting an instruction to the pipeline to change urgency of the thread. Keckler has no teaching or suggestion of modifying a thread's urgency indicator by inserting an instruction to the pipeline, as required by claim 11.

Claim 12 recites deactivating the first thread and activating a second thread, and modifying urgency of the second thread. As argued above, Keckler does not modify a thread's urgency. Further, Keckler certainly does not disclose inserting an instruction into the pipeline to modify an threads urgency.

Claim 13 recites monitoring possible switch points of an inactive thread having a second urgency indicator that is based upon expected progress of the inactive thread within the pipeline execution units, and deactivating the first thread, or not, based upon a first and second urgency indicators. As argued above, the priority system of Keckler is not equivalent to the urgency indicator of the immediate application. The priority system of Keckler is static and is not based upon expected progress of a thread. The priority system of Keckler provides a selection decision between threads based upon a desired progress and not an expected progress. Urgency indicators of the immediate application are based upon determined progress of a thread though a pipeline. In the priority system of Keckler, a high priority thread may stall frequently and therefore have less throughput than a low priority thread that does not stall. Thus, Keckler's priority is not equivalent to urgency of the immediate application.

For at least these reasons, Keckler cannot anticipate claims 2-13. Reconsideration of claims 2-13 is respectfully requested.

Claim 14 recites a processor for processing multi-threaded program instructions, including:

i. an array of pipeline execution units and associated heuristics affecting how the instructions are processed within the units; and

ii. a thread controller for monitoring processing of the instructions within the units and for switching between multiple program threads based upon (a) the heuristics and (b) urgencies of the program threads;

iii. wherein the urgencies are based upon one or both of (a) progress of the threads through the pipeline execution units and (b) expected progress of the program threads through the pipeline execution units.

Keckler does not disclose or suggest a thread controller for monitoring processing of the instructions within pipeline execution units and for switching between multiple program threads based upon heuristics and urgencies of the program threads, as required by element ii of claim 14. Instead, and as noted above, Keckler's multithreading is implemented by interleaving instructions from different threads over the execution resources of each cluster on a cycle-by-cycle bases. Keckler specifically teaches away from claim 14 by reciting that "unlike block multithreading [1], [34], which switches threads on long latency operations, ... the MAP chip makes its thread selections based upon the availability of an instruction's register operands." See Keckler page 908, section 3.2.

By way of comparison, multithreading within the immediate application switches threads, or not, based upon one or more threads 'urgencies', which may be based upon "a thread instruction missing the cache" or "processor interrupts" (see paragraphs [0006] and [0007]). As argued above, Keckler's thread priority is not equivalent to thread urgency of claim 14. Also as argued above, in Keckler, indication of whether all of the data for the instruction has arrive is not equivalent to an urgency indicator as required by claim 14. Again, as argued above, thread priority of Keckler does not indicate expected progress of a thread; it merely gives a priority ordering of thread selection and is not based upon expected throughput of the thread; a high priority thread can still stall in the pipeline and have lower throughput than a low priority thread.

Reconsideration of claim 14 is requested.

Claims 15-19 depend from claim 14 and benefit from like argument; but in addition, these claims also have features that are patentably distinct from Keckler. For example, claim 15 recites one or more of time slice expiration heuristics, cache miss heuristics and processor interrupt heuristics. Keckler does not disclose or suggest time slice expiration heuristics as in claim 15. As argued above, the preempt state of Keckler is not equivalent to a time slice.

Claim 16 recites one or more instructions, one of the instructions changing urgency for at least one thread of the processor. Keckler does not disclose or suggest program threads with one of the instructions changing urgency of at least one thread of the processor, as in claim 16.

Claim 17 recites the controller modifying an urgency of any of the threads to modify future treatment of the threads in switch out events. Keckler does not disclose or describe a controller modifying an urgency of any of the threads to modify future treatment of the threads in switch out events. And Keckler does not disclose that priority of a thread is modified, nor urgency of a thread. In paragraph 21 of the pending office action, the Examiner asserts that after 255 cycles, the urgency of the Keckler's instruction is modified in order to allow the thread to execute. But, Keckler does not disclose or use urgency. As argued above, the priority of Keckler is not equivalent to urgency as used in claim 17. Keckler further does not disclose modifying priority to allow a thread to execute.

Claim 18 recites the controller either decreasing or increasing urgency for the program threads by injecting an instruction to the pipeline execution units. Nowhere within Keckler is there disclosure of a controller decreasing or increasing the urgency (or priority) for the program threads by injecting an instruction into the pipeline execution units, as required by claim 18.

Claim 19 recites a time slice expiration unit for monitoring expiration of threads within the processor. Clearly, Keckler does not disclose or suggest a time slice expiration unit for monitoring expiration of threads within the processor, as required by claim 19, since, as argued above, Keckler does not operate with time slicing.

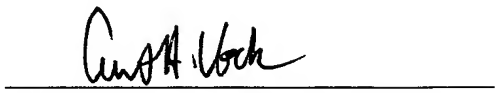
At least for the above reasons, Keckler cannot anticipate claims 15-19. Reconsideration of claims 15-19 is respectfully requested.

In view of the above arguments, claims 1-19 patentably distinguish over Keckler. Generally, Keckler describes only how to select which thread becomes active at an event. In accord with the inventions of claims 1-19, on the other hand, active and inactive events may be examined and may or may not switch based on determination of urgency.

Claims 1-19 are therefore deemed allowable. Should the Examiner disagree, we ask for the opportunity to interview.

It is believed that no fees are due in connection with this amendment. If any fee is due, please charge Deposit Account No. 08-2025.

Respectfully submitted,

By: 

Curtis A. Vock, Reg. No. 38,356
LATHROP & GAGE L.C.
4845 Pearl East Circle, Suite 300
Boulder, CO 80301
Telephone: (720) 931-3011
Facsimile: (720) 931-3001